Nature-Inspired Optimisation Approaches and the New Plant Propagation Algorithm

Conference Paper · June 2011

CITATIONS
95

READS
1,124

2 authors:

Abdel Salhi
University of Essex
129 PUBLICATIONS **1,324** CITATIONS
SEE PROFILE

Eric Fraga
University College London
182 PUBLICATIONS **2,530** CITATIONS
SEE PROFILE

Proceeding of The International Conference on Numerical Analysis and Optimization (ICeMATH2011)
Keynote Speaker - Abdellah Salhi

Nature-Inspired Optimisation Approaches and the New Plant Propagation Algorithm

Abdellah Salhi
Department of Mathematical Sciences
The University of Essex
Colchester CO4 3SQ
as@essex.ac.uk

Eric S Fraga
Centre for Process Systems Engineering
Department of Chemical Engineering
UCL (University College London)
e.fraga@ucl.ac.uk

***Abstract*—Nature-inspired algorithms are proving to be very successful on complex optimisation problems. A new algorithm, inspired by the way plants, and in particular the strawberry plant, propagate is presented. The algorithm is explained, tested on standard test functions, and compared with the well known Nelder-Mead algorithm. The new approach is then applied to a complex process design problem that arises in Chlorobenzene purification, a problem that exhibits strong nonlinear behaviour and has a small feasible region.**

***Index Terms*—Optimisation, Optimisation methods, Pareto optimisation, Process design.**

I. Introduction

Since the advent of Simulated Annealing of Metropolis *et al.* [1] and the Genetic Algorithm of Holland [2], a flurry of nature-inspired algorithms has been introduced with increasing successes on the most challenging problems in nonlinear and combinatorial optimisation. These successes are particularly notable when comparing against classical gradient-based approaches to optimisation.

Nature-inspired algorithms and heuristic procedures have some notable drawbacks; unlike classical approaches, they have a rather limited theory to support them. They also are mainly stochastic, and dependent on often many parameters the setting of which is usually arbitrary. This means that results change from run to run and replication of results by different researchers may be hard to achieve.

It is also true that of the current nature-inspired algorithms, few, if any, are inspired by plants. Consider for a moment the way plants propagate. Given their resilience, their ability to colonise new territories in search of favourable growing conditions, they must have developed effective propagation strategies. In the following we report on a particular plant, namely the strawberry plant, observe its propagation strategy and see how an effective algorithm can be designed based on this strategy.

II. The strawberry plant approach to survival through an adapted propagation strategy

The strawberry plant (*Fragaria X ananassa*) [3] belongs to the Rose family. The strawberry-growing industry started in Paris in the seventeenth century with the European variety. In 1714, Amedee-Francois Frezier, a mathematician and engineer, hired by Louis XIV to draw maps of South America returned from Chile with some Chilean strawberry plants which give a larger fruit. Subsequent crossings and selections led to the modern plant.

A. Propagation strategy

Seeds appear on the fruit and, in principle, the plant can propagate using them. This should be true for the pure varieties; hybrids, like the modern ones, are generally infertile. Instead, these plants use *runners* to propagate.

The *maiden* plant sends runners which, upon touching the ground, grow roots from which daughter plants emerge. The question of interest to an optimisation audience is whether the mother plant sends runners in a totally random fashion or if there is an underlying principle or strategy that the plant follows to optimise its propagation and ultimately its survival.

If a strawberry plant is left alone to grow as in the wild, one can observe, after a period of time of two to three seasons that strong and well-established plants have a concentration of younger plants around them. On the other hand, those plants which are less established and do not look very strong send few but long runners.

Although runners are sent in all directions, there is a concentration toward areas with better lighting and humidity. This must be due to what, in the botanical parlance, is called *tropism* or growth response to stimuli [4].

B. Observations and assumptions

It is assumed, therefore, that the strawberry plant, as well as other plants, have an underlying propagation strategy. This strategy is developed over time to ensure the survival of the specie. Put crudely, a plant "will strive" to have its offspring in areas of the ground that offer the necessary nutrients and growth potential. When a plant is in a good spot it will send short runners in large numbers; when it is in a poor spot it will send long runners in search of better spots. The long runners are in small numbers since they are an investment and, being in a poor spot, the plant may not have sufficient resources to send many such runners.

With these observations and assumptions in mind, it can be stipulated that the strawberry plant, and other plants, in order to thrive in a given environment, solve, in effect, a *survival optimisation* problem. The approach we put forward

Require: objective $f(x)$, $x \in \mathcal{R}^n$
  Generate a population $P = \{p_i, i = 1, \ldots, m\}$
  $g \leftarrow 1$
  for $g \leftarrow 1$ **to** $g_{\max}$ **do**
    compute $N_i = f(p_i), \forall p_i \in P$
    sort P in descending order of N
    create new population ϕ
    for each $p_i, i = 1, \ldots, m$ **do** {best m only}
      $r_i \leftarrow$ set of runners where both the size of the set and the distance for each runner (individually) is proportional to the fitness N_i
      $\phi \leftarrow \phi \cup r_i$ {append to population; death occurs by omission above}
    end for
    $P \leftarrow \phi$ {new population}
  end for
  return P, the population of solutions

Fig. 1. The plant propagation algorithm (PPA).

here consists in mapping an optimisation problem onto the survival optimisation problem of the strawberry plant and in adopting its strategy for survival in the environment to look for points in the search space that give our objective function good values and ultimately the best value.

C. Mapping an optimisation problem to the survival problem of the strawberry plant

Consider the optimisation problem

$$\max_{x \in S} z = f(x)$$

where S is the search domain, often described by box constraints but which may also include other constraints. The strawberry plant survival problem is to find the best spot x in the plot S which will give best growth $f(x)$ to the offspring of the plant.

III. The Plant Propagation Algorithm (PPA)

There are two characteristics that permeate all successful algorithms/heuristics for global optimisation: concentration or intensification and diversification or exploration. Concentration allows to search locally and converge to a local optimum while diversification allows the search to avoid getting trapped in the attraction region of a local optimum, giving the algorithm the potential to find the global optimum. These two characteristics are conflicting and the success of any implementation of a search algorithm will depend on the balance between the two characteristics.

The strawberry approach implements concentration by sending many short runners from *good* solutions. It implements diversification by sending fewer but longer runners from those solutions which are less good. The full algorithm is presented in Figure III.

Like most algorithms of this nature, the PPA requires customisation through the definition of some utility functions and the assignment of values to a number of a parameters. For the PPA, these are as follows: the population size, the number of generations, a *fitness* function, the number of runners to create for each solution and the distance for each runner.

The PPA is based on a population of shoots, each of which represents a solution in the search space. Each shoot is assumed to have taken root which is equivalent to the objective function being evaluated. Each shoot will subsequently send out runners to explore the solution space around it. The number of shoots is given by this parameter and is indicated by m in the algorithm.

The algorithm is iterative with all shoots sending out runners at each generation. This parameter provides a terminating criterion based on how many times to send out runners and is represented by $g_{\max}$.

Solutions in the population will be sorted according to their *fitness*. This fitness will naturally be a function of the objective function values but the actual association between objective function value and fitness may be tailored for the specific problem addressed. It is, however, assumed by the algorithm presented that the fitness values, $f(x)$, will satisfy $f(x) \in [0, 1]$; if not, the equations used for determining the number of runners and the distance for each runner to run should be modified. The actual fitness functions used for the case studies presented below will be defined along with the problem statements.

The functions used to determine the number of runners and the distance each runner should travel are defined below. They require that the fitness values lie strictly in $(0, 1)$. We map the fitness values, $f(x)$, described above to ensure this condition:

$$N(x) = \frac{1}{2}\left(\tanh\left(4f(x) - 2\right) + 1\right) \tag{1}$$

The effect of this mapping function is shown in Figure 2. The specific need for this mapping is discussed below. However, it also provides a means of emphasising further better solutions over those which are not as good.

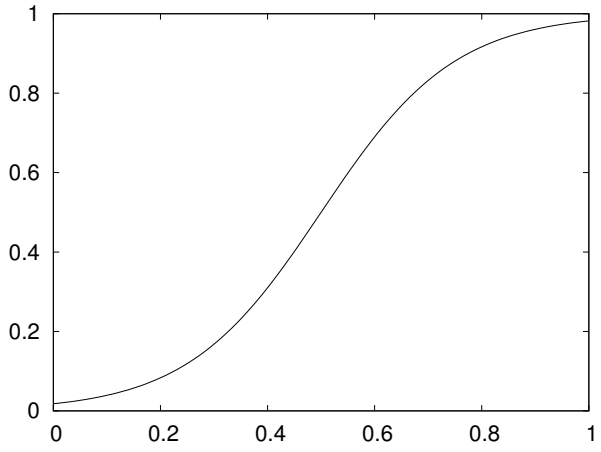

Fig. 2. Mapping function to convert fitness values in $[0, 1]$ to $(0, 1)$ while emphasising better solutions.

The number of runners generated by a solution should

be proportional to its fitness. By default, we have used the following function:

$$n_r = \lceil n_{\max} N_i r \rceil \tag{2}$$

where n_r is the number of runners to generate for solution i in the current population, after sorting, $n_{\max}$ is the maximum number of runners to generate, N_i is the mapped fitness (using equation 1 above) of solution i, and $r \in [0, 1]$ is a randomly chosen number for each individual solution at each generation. The combination of the fitness mapping function and the ceiling operator ensures that every solution will generate at least one runner, even for the least fit solutions, ones with $f_i(x) \equiv 0$. The most fit solutions will generate at most $n_{\max}$ runners. For all case studies presented in this paper, $n_{\max} = 5$.

The distance each runner will cover follows a similar rule although one where it is inversely proportional to the fitness, as described above. This distance is:

$$d_{r,j} = 2(1 - N_i)(r - 0.5) \tag{3}$$

for $j = 1, \ldots, n$, where n is the dimension of the search space. Each $d_{r,j}$ will be in (-1,1). The fitness mapping function ensures that the best solutions will have the potential to send out runners some distance > 0 even if $f_i(x) \equiv 1$. The distance calculated will be used to update the solution i based on the bounds on x_j:

$$x_j^* = x_j + (b_j - a_j) d_{r,j} \tag{4}$$

These x_j^* values are then adjusted to ensure that the new point generated is within the bounds $[a_j, b_j]$. Given equation 3, there will be some preference for points being generated at the bounds of the search space. For chemical engineering process design problems, such as the main case study presented below, this is actually a useful property; for more general problems, this is likely to be a slight disadvantage at best.

IV. Computational results on standard test functions

Before attempting a problem from the chemical engineering domain, we evaluated the proposed method on a set of functions over domains described by box constraints [5], [6].

Table I presents the results of our implementation of the PPA. For each problem, the method has been run 10 times. In all cases, except as noted below, we have used the following values for the parameters of the PPA: $g_{\max} = 30$, $m = 30$, and $n_{\max} = 5$. The fitness function is defined as

$$f(z) = \frac{z_{\max} - z}{z_{\max} - z_{\min}}$$

where $z_{\min}$ and $z_{\max}$ are the minimum and maximum objective function values in the current population and our objective is to minimise; if all solutions in the population have the same objective function value, all solutions are given a fitness of 0.5.

The best result obtained, over the 10 runs, is presented, in terms of objective function value, as well as the percentage of the time that the method is able to find the optimum point, to within 1% of the box size from the location of the known theoretical optimum. Only one problem, Easom, poses a significant challenge to this new method although, even in this case, the PPA method finds the optimum half of the time. The PPA is successful almost always for all the other problems.

TABLE I
Summary of results obtained with the novel PPA method. Ten attempts were made for each problem and solutions obtained were compared with the theoretical optimal solution known. a and b are the lower and upper bounds on x, the decision variables.

Problem	a	b	x_0	Best objective	Success rate (%)
Six hump camel back	-3 -2	3 2	1 1	-1.0316	80
Branin	-5 -5	15 15	0 0	-0.3980	100
Easom	-100 -100	100 100	-1 1	-0.9997	50
Goldstein Price	-2 -2	2 2	1 1	3.0002	90
Martin Gaddy	-20 -20	20 20	0 0	0.0000	100
Rastrigin	-10 -10	10 10	-3 2	0.0092	90
Rosenbrock	-5 -5	10 10	0 0	0.0002	100
Schwefel	-500 -500	500 500	0 0	-837.9650	70

It should be noted that these same problems have been attempted with the Nelder-Mead simplex method [7], using the implementation provided by Kelley [8] using the same initial guesses. In all but two problems, the PPA was able to find a better solution, often significantly better. In the two exceptions, the methods were equivalent.

From the table, we see that the PPA method is able to find solutions close to the optima for all problems. It performs significantly better than the Nelder-Mead method in all cases. This gave confidence in the ability of this method to perform well on more challenging problems. It has, thus, been tried on an optimisation problem from an industrial application.

V. Industrial application case study 3: Chlorobenzene purification process design

The design and optimisation of process flowsheets is a challenging task due to the nonlinear models required and the multi-criteria nature of the objectives for evaluating alternative designs. A process flowsheet consists of a number of processing steps, often referred to as *units*, with *streams* connecting the steps. The problem of design is to determine the operating

TABLE II
FEED STREAM DEFINITION FOR THE BENZENE RECYCLE CASE STUDY.

	Component		Flow ($\frac{kmol}{s}$)
1.	Benzene	C_6H_6	0.97
2.	Chlorobenzene	C_6H_5Cl	0.01
3.	Di-Chlorobenzene	p-$C_6H_4Cl_2$	0.01
4.	Tri-Chlorobenzene	$C_6H_3Cl_3$	0.01
	Pressure	1 atm	
	Temperature	313 K	

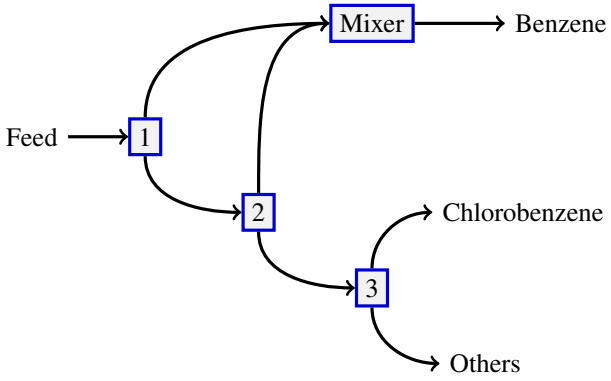

Fig. 3. Process structure for the chlorobenzene purification stage showing three distillation units (numbered) with the desired product streams, Benzene and Chlorobenzene, and one waste stream, "Others".

conditions and sizing parameters for the units in the process to achieve desired objectives subject to a number of constraints, both physical (or chemical) and economic.

The case study presented here is the optimisation of a purification section of a process for the production of chlorobenzene. In the overall process, large quantities of benzene are used. Due to the partial conversion in the reaction section of the process, significant quantities of un-reacted benzene could be wasted if not recycled. To recycle the benzene, a stream consisting of primarily benzene needs to be further purified to ensure that the benzene sent back upstream in the process is pure enough to not affect the reaction. The case study we consider here is the design of this purification stage of the overall process.

The stage comprises three distillation units with a feed stream described in Table II. Although the main aim is to purify the benzene recycle stream, the feed stream to this stage also contains some amount of the main product of the process, chlorobenzene. It is desirable to separate this product as well in sufficiently pure form and to avoid unnecessary loss. The process structure we consider is presented in Figure 3.

A. Process modelling

To assess the different alternatives, economic criteria in the form of capital and operating costs are used. Process models are required to determine the impact of design decisions on these criteria. For this problem, the process models consist of the Fenske, Underwood and Gilliland short-cut correlations often used for multi-component distillation column design [9].

These models are based on the concept of *light* and *heavy* key species. Essentially, if the species in a mixture are sorted according to their boiling points, the mixture can be separated between any two adjacent species. For instance, if there were a mixture with three species, A, B, and C, with increasing boiling points from A to C, there would be two possible separations for this mixture. The first would be between A and B, yielding two streams, one with almost all of the A but a little B and one with a little of the A, most of the B, and essentially all of the C. The second separation possible would be between B and C: one of the resulting streams from this separation would have all the A, most of the B and a little C and the other stream would have a little B and most of the C.

The Fenske equation determines the minimum number of stages, $N_{\min}$, a distillation column will require to achieve a given separation identified by the recovery of the *light* (most volatile component in the separation) and the *heavy* (least volatile of the components):

$$N_{\min} = \frac{\log \frac{x_{D,l} x_{B,h}}{x_{B,l} x_{D,h}}}{\log \alpha_{l,h}}$$

where D refers to the *distillate* or tops product of the distillation unit and B to the *bottoms* product. x is the molar composition of each species in the streams, $\alpha_{l,h}$ is the relative volatility of the light key to the heavy key. The compositions of the light and heavy keys in the separation are a function of one of the design variables, the *recovery* desired. The higher the recovery, the smaller the amount of each key that goes out with the other stream so the lower the potential losses. However, more stages are required for higher recoveries.

The Underwood equation determines the minimum amount of *reflux*, the liquid sent back into the column from the top stream to aid in achieving the separation:

$$R_{\min} = \sum_{i=1}^{n_c} \frac{\alpha_i x_{D,i}}{\alpha_i - \theta}$$

where θ is the solution to the nonlinear equation

$$\sum_{i=1}^{n_c} \frac{\alpha_i x_{F,i}}{\alpha_i - \theta} = 1 - q$$

where F indicates the feed stream, n_c is the number of species involved in the feed stream and q is a function of the *state* of the feed stream. The state depends on the energy in the stream and that is a function of the temperature and the pressure of the stream.

The actual number of stages and the actual reflux used are correlated through the Gilliland equation. This is also a nonlinear equation and depends on another design variable, the reflux rate factor. The higher this value, the larger the actual reflux rate used. The reflux rate affects the operating costs through the use of cooling and heating utilities: larger reflux means increased utility use). It also influences the capital cost: higher reflux leads to a wider column but one with less stages, usually leading to a lower capital cost overall.

In all of the above equations, the calculation of variables such as α and q require the estimation of physical properties. Physical property models are typically nonlinear. For this problem, we have used the Antoine equation to determine vapour pressures, p^* used to calculate the relative volatilities, α, of the species:

$$\log p^* = A - \frac{B}{T - C}$$

where A, B and C are the *Antoine coefficients* which can be found in reference books for many species. The relative volatilities of species are the ratio of their vapour pressures. The temperatures will depend on the operating pressure, the third design variable for the distillation unit. Higher pressures will results in higher temperatures and smaller relative volatilities. The former will affect the choice of heating and cooling utilities; the latter will affect the number of stages required for the separation required.

The process design has several constraints beyond the physical models described above. Firstly, the benzene and chlorobenzene product streams must meet purity requirements. Secondly, there is a finite number of utilities available for meeting the heating and cooling requirements and, for some combinations of operating pressure and stream compositions, utilities may not be available to meet the requirements imposed by the design of the distillation units. Finally, there is a limit on how much of the main product, chlorobenzene, can be lost through the bottom product stream of the 3rd distillation unit. The constraints define a feasible region that is small, well under 1% of the box domain defined by the design variable bounds and, together with the nonlinearities, make the problem difficult to solve. The full model is further described elsewhere [10].

B. Optimisation problem

The two economic criteria, operating and capital costs, are conflicting and the engineer will be interested in the trade-off between them due to uncertainty in energy markets and product prices. The goal is to identify a suitable trade-off, or Pareto, curve for this problem. The minimisation problem, therefore, is formulated as

$$\min_d z = \lambda c_c + (1 - \lambda) c_o \tag{5}$$

where c_c is the capital cost, c_o is an annual operating cost, $\lambda \in [0, 1]$ and d are the design variables. For the 3 distillation unit process, there are 9 design variables: operation pressure, $P \in [1, 32]$, recovery, $r \in [0.8, 0.999]$, and reflux rate factor, $R \in [1.05, 1.5]$, for each distillation unit. We solve this problem for different values of λ to generate an approximation to the Pareto front.

The capital costs depend on the sizes of the individual units. Cost models from Rathore et al. [9] are used. Operating costs include the cost of heating and cooling utilities, from a discrete set of possible choices, and maintenance costs which are proportional to the capital costs. The operating cost models are also from Rathore et al. [9].

C. Applying the PPA

Due to the possible presence of infeasible solutions in the population used by the Strawberry algorithm, a new fitness function has been defined. There are three cases: all solutions in the population are feasible, all are infeasible and there is a mix of feasible and infeasible solutions. For the first two, the fitness function is as defined previously:

$$f = \frac{\max - y}{\max - \min}$$

where y is either the actual objective function value for feasible solutions or the constraint violation for infeasible ones. For the last case, the same mapping function is used but compressed and shifted so that all feasible solutions are given fitness values $\in [0.5, 1]$ and infeasible solutions $\in [0, 0.5]$. The procedure is otherwise the same as described above, including the use of the fitness mapping function (eq. 1). A population size of $m = 30$ has been used, allowing the method to run for up to $g_{\max} = 500$ generations.

The solutions obtained solving this problem with $\lambda = 0, 0.05, 0.1, \ldots, 1$ with ten attempts for each value of λ are presented in figure 4. This figure contains the union of the populations at the end of each of the runs. Two conclusions can be drawn from this figure:

1) There appears to be little spread in the results obtained by different runs, evidenced by the small depth of the front. The solutions obtained all appear to converge to points on the front as identified by the union of all of the solutions. This can be further illustrated by comparing the plot of all the points in Figure 4 with the plot, in Figure 5, of only those points which are not dominated.
2) There are gaps in the Pareto front. These are not an artefact of the solution method but are actually a real property of the solution. The discrete nature of utilities means that there is a discontinuity in the operating cost as a function of the pressure design variable. The gaps correspond to the choice of different utilities. Moving along the front from left to right, cheaper utilities become available as the operating pressure changes.

For design problems, the feasible space may be a small fraction of the domain defined by bounds on the design variables. This is the case for this example. Often, even finding a feasible point may be considered a success for an optimisation method. Figure 6 shows the evolution of the best solution in the population for a given instance of the problem. The left axis is for a measure of constraint violation; feasible solutions require this value to be 0. Initially, all solutions in the population are infeasible. There is a quick reduction in the measure of constraint violation followed by a steady and smooth decrease to a feasible solution. Once a feasible solution is found, the plot shows the objective function value of the best solution in the population; the population will likely contain infeasible solutions for some time. The right axis shows the value of the feasible solutions. Again, we see a quick reduction in objective function value from the point where a feasible

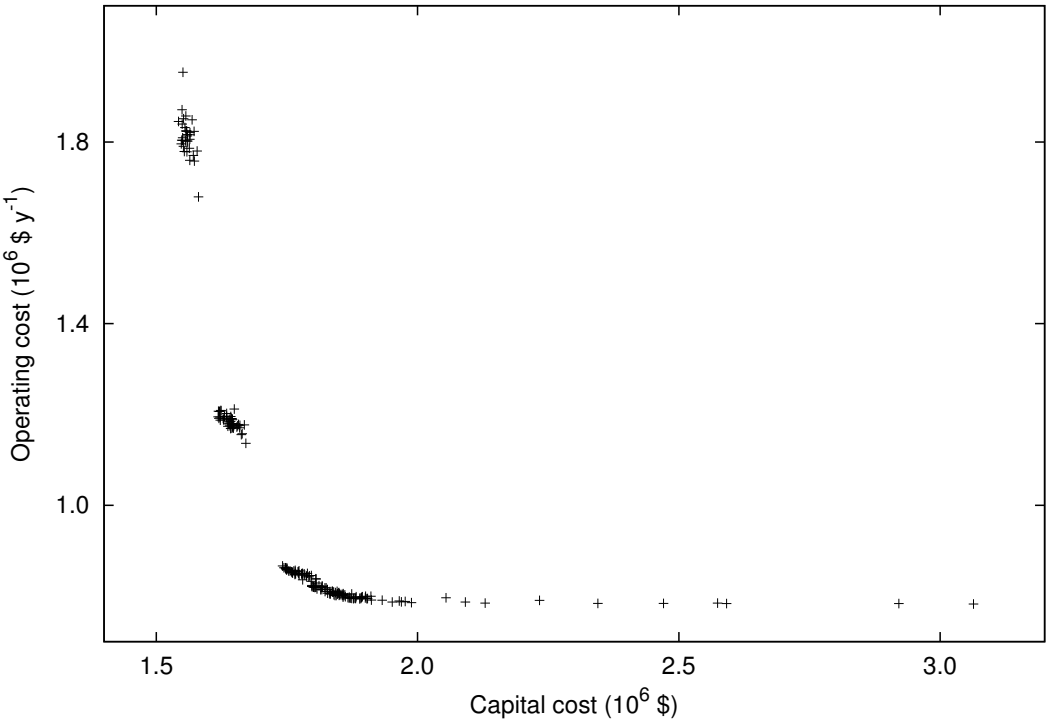

Fig. 4. The full set of points returned by the various runs of the method for different values of $\lambda \in [0, 1]$, showing an approximation to the Pareto front for the bi-criterial optimisation problem.

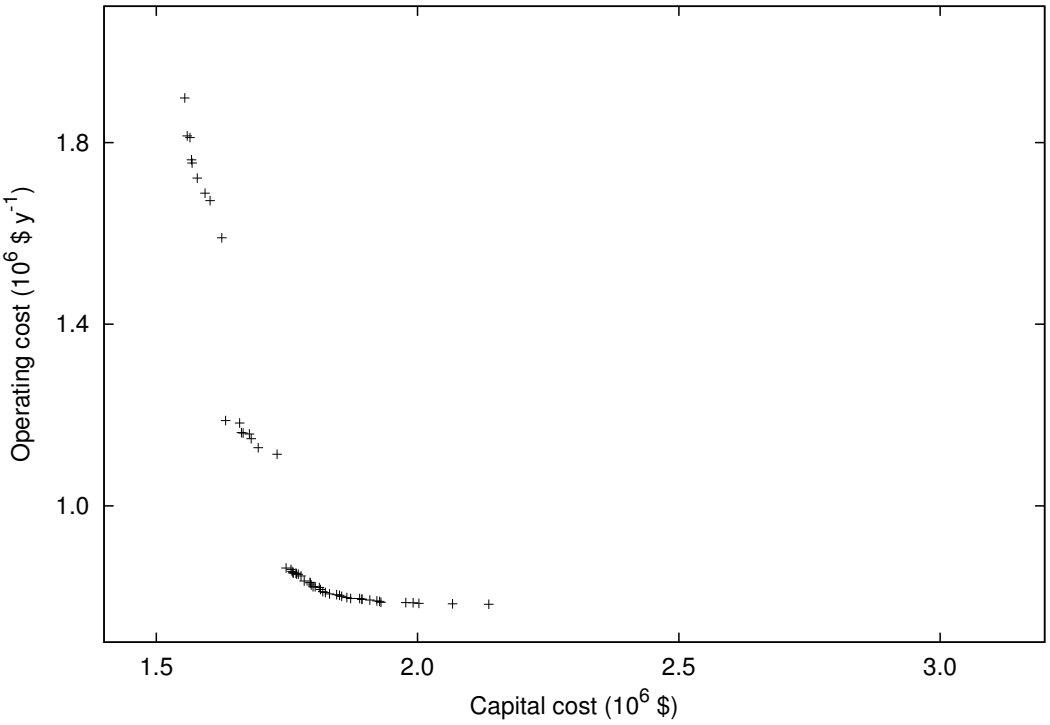

Fig. 5. The subset of non-dominated points from the set of all points showing the approximation to the Pareto front.

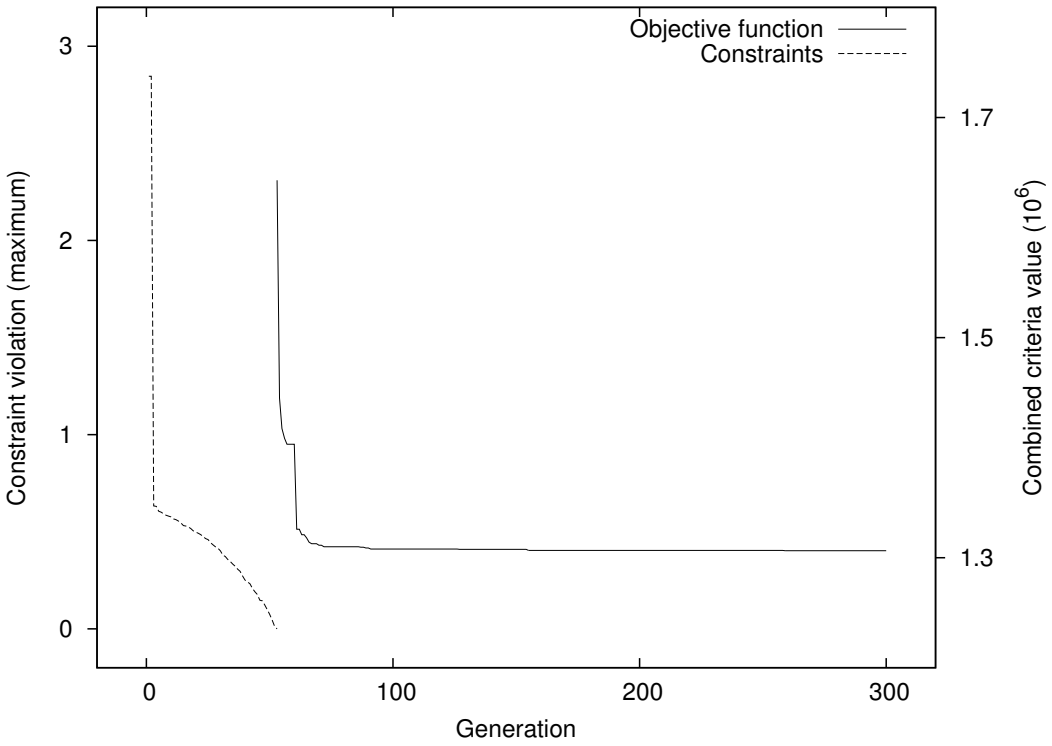

Fig. 6. Graph of the solution in a population as it evolves. The population may include points at which the design is infeasible. Initially, until generation 52, all solutions in the population are infeasible with the measure of constraint violation on the left y axis. At generation 53, a feasible point has been found and the best solution in all subsequent generations will be feasible with a value indicated by the right y axis.

solution is identified and then a steady decrease until the method appears to converge at around 100 generations. For this example, a population size of 30 was used.

VI. Conclusion

A new family of stochastic algorithms for optimisation has been described, based on the propagation of plants, with the presentation of a specific implementation emulating the behaviour of the strawberry plant. The new method implements the two most important characteristics of successful global optimisation algorithms namely diversification and concentration or exploration and intensification. Diversification/exploration is implemented through sending long runners from plants which are not placed in profitable spots; concentration/intensification is implemented through sending short runners from plants which are in good/profitable spots.

The concepts of good *spots* and *long* or *short* runners are captured, in this new population-based method, with the concepts of *fitness* and *distance*, respectively. Furthermore, the concept of fitness is key to the definition of the two plant propagation elements: generating runners to propagate and determining the distance each runner travels. The implementation has been tested on a number of test cases from the literature and applied to a complex problem in chemical plant design.

The results show that the method is effective in identifying the global optimum in all of the cases considered. It performs significantly better than the Nelder-Mead direct search method, a useful benchmark for comparison. Although there is a set of parameters for the method, the same values of these parameters appear to work effectively for a wide range of problems. In other words, they appear to be robust default values.

The method has been shown to solve the industrial case study with good convergence to the bi-criterial trade-off curve. Through the definition of an appropriate fitness function, the method is able to handle infeasible solutions. This is critical for many design problems where the feasible region may be small in comparison with the full search domain. The PPA method is able to find feasible points quickly and then evolve towards the best solutions steadily.

Although the parameter values chosen in this paper appear to be suitable for the problems investigated, little analysis has been performed to understand the impact of these parameters on the efficiency of the method. This is planned for future work. A second aspect we intend to consider is the incorporation of Pareto specific fitness functions that will allow the method to target multi-criteria problems directly instead of through a weighting function of objectives. It is likely that we will consider a fitness function similar to that used in [11], based in turn on the work of Deb [12].

References

[1] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[2] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press, 1975.

[3] A. C. Gibson, "Writups and illustrations of economically important plants," http://www.botgard.ucla.edu/html/botanytextbooks/economicbotany/Fragaria/index.html.

[4] S. Gilroy and P. H. Masson, *Plant Tropisms*. Blackwell Publishing, 2008.

[5] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European Journal of Operational Research*, vol. 185, pp. 1155–1173, 2008.

[6] H. Pohlheim, "Examples of objective functions," 2005, in *Documentation for Genetic and Evolutionary Algorithms for use with MATLAB: GEATbx version 3.7*, http://www.geatbx.com.

[7] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.

[8] C. T. Kelley, *Iterative methods for optimization*. SIAM, 1999.

[9] R. N. S. Rathore, K. A. van Wormer, and G. J. Powers, "Synthesis strategies for multicomponent separation systems with energy integration," *AIChE Journal*, vol. 20, no. 3, pp. 491–502, 1974.

[10] A. Žilinskas, E. S. Fraga, and A. Mackutė, "Data analysis and visualisation for robust multi-criteria process optimisation," *Computers and Chemical Engineering*, vol. 30, no. 6-7, pp. 1061–1071, 2006.

[11] G. Fiandaca, E. S. Fraga, and S. Brandani, "A multi-objective genetic algorithm for the design of pressure swing adsorption," *Engineering Optimization*, vol. 41, no. 9, pp. 833–854, 2009. [Online]. Available: http://www.informaworld.com.libproxy.ucl.ac.uk/10.1080/03052150903074189

[12] K. Deb, "An efficient constraint handling method for genetic algorithms," *Comput. Methods Appl. Mech. Engng.*, vol. 186, pp. 311–338, 2000.